Project :Omega – Tutoriel

Introduction à F-Script

Auteur : Philippe Mouguin - pmougin@acm.org
Traduction : Pejvan BEIGUI
pejvan@projectomega.org

SOMMAIRE

INTRODUCTION

Une application importante de l'ouverture de Cocoa est son intégration avec des langages de script tels que Python, JavaScript, Lua, Ruby et Guile. Il existe plusieurs packages qui fournissent cette intégration, que cela soit pour Mac OS X ou pour GNUStep. Ces outils sont très utiles parce qu'ils permettent une utilisation intéractive et une insertion facile des composants de Cocoa.

F-Script par Philippe Mougin est un projet open-source sur Mac OS X qui a attiré notre attention. Cette fine couche de scripting orienté objet permet d'établir une interaction avec les frameworks Cocoa et les propres objets du développeur. Nous le présentons ici comme partie intégrante de notre exploration des outils se rapportant à Cocoa.

Les articles

Scripter Cocoa avec F-Script

Dans cet article, Philippe Mougin nous donne un avant goût du scripting de Cocoa et nous montre le niveau d'intégration que l'on peut en attendre. Vous utiliserez F-Script, un langage de scripting open-source pour Cocoa, qui permet la création d'applications graphiques. En utilisant F-Script, vous allez programmer en attaquant directement l'Application Kit, qui est le framework orienté objets de Cocoa pour les interfaces graphiques. *11 Septembre 2002*

SCRIPTER COCOA AVEC F-SCRIPT

par **Philippe Mougin** le 30/11/2001
traduit par **Pejvan BEIGUI** le 08/09/2002

***Note de l'éditeur** : Nous voyons beaucoup d'agitations de la communauté open-source autour de Mac OS X. Un des projets qui m'a tapé dans l'oeil est celui de Philippe Mougin : F-Script. Cette couche de scripting, orientée objets et légère, fournit un accès interactif aux frameworks Cocoa et aux objets personnalisés. Je vous présente ce tutoriel comme un élément de notre exploration de Cocoa et des outils qui s'y apparentent. [...]*

Grâce à Objective-C, Cocoa est construit sur un modèle objet dynamique, réfléchi et ouvert. Ceci rend notamment possible la création d'éditeurs visuels de graphes d'objets comme Interface Builder, d'outils de surveillance de l'environnement d'exécution, d'outils de navigation parmi les objets et de nombreux autres outils tirant profit des capacités de l'environnement d'exécution de Cocoa.

Une application importante de l'ouverture de Cocoa est son intégration avec des langages de script tels que Python, JavaScript, Lua, Ruby et Guile. Il existe plusieurs packages qui fournissent cette intégration, que cela soit pour Mac OS X ou pour GNUStep. Jetez un oeil, à titre d'exemples, à Joy de AAA+, Lua Objective-C de Steve Dekorte, RIGS de Laurent Julliard, GNUstep Guile de Richard Frith-MacDonald et enfin StepTalk de Stefan Urbanek. Ces outils sont très utiles parce qu'ils permettent une utilisation intéractive et une insertion facile des composants de Cocoa.

Dans cet article, je voudrais vous donner un avant goût du scripting de Cocoa et vous montrer le niveau d'intégration que vous pouvez en attendre. Nous utiliserons F-Script, un langage de scripting open-source pour Cocoa, qui permet la création d'applications graphiques. En utilisant F-Script, nous allons programmer en attaquant directement l'Application Kit, qui est le framework orienté objets de Cocoa pour les interfaces graphiques.

Remarquez que l'on peut construire ce genre d'application en utilisant Interface Builder, mais dans cet article, notre but est d'illustrer le scripting de Cocoa et donc nous utiliserons l'Application Kit programmaticallement.

Hé ! Un autre convertisseur de devises

Notre application est un convertisseur de devises qui ressemble à celui que l'on trouve dans les tutoriels d'Apple (reportez vous ici).

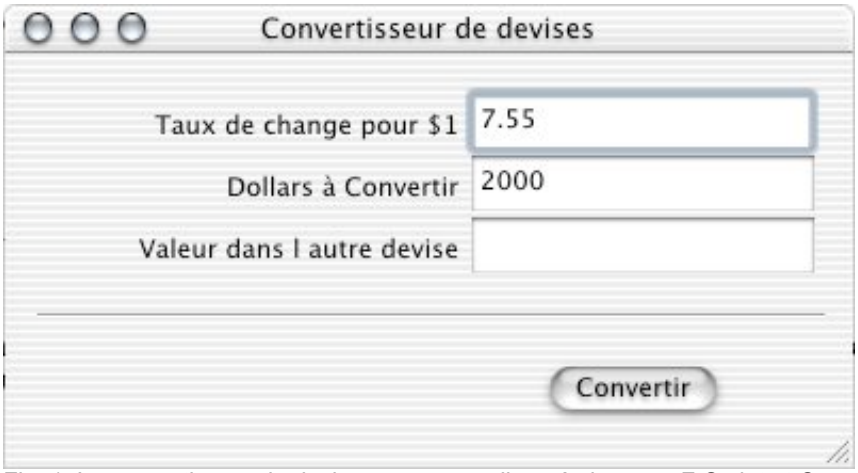

Fig. 1. Le convertisseur de devises que nous allons écrire avec F-Script et Cocoa.

F-Script

Vous pouvez récupérer F-Script sur fscript.org. Dans cet article, nous utilisons F-Script 1.1 sur Mac OS X.

F-Script est un langage de scripting purement orienté objets avec des concepts et une syntaxe similiares à ceux de SmallTalk. Vous pouvez interagir avec F-Script au travers de l'application "`fs`" qui vous fournit un shell interactif dans lequel vous pouvez rentrer vos instructions.

Quelques éléments de base de la syntaxe d'F-Script :

- Les instructions sont séparés par "."
- ":=" dénote une affectation.
- les chaines de caratères (Strings) utilisent les guillements simples : 'Une chaine'.
- les commentaires utilisent des guillements doubles : "Un commentaire"

Le script du convertisseur de devises, version 1

Le script suivant peut être inséré dans le shell d'F-Script par un simple copier/coller. Vous pouvez aussi rentrer les instructions pas à pas : vous verrez alors les éléments de l'interface du convertisseur de devises apparaître petit à petit. Bien entendu, vous pouvez modifier les valeurs des arguments utilisés dans ce script, ou en omettre certains pour voir ce qui se produit. Vous pouvez même interagir avec les objets une fois que vous avez construit le convertisseur de devises.

Convertisseur de devises F-SCRIPT (version 1)

```
    "Instantiation et configuration d'une fenêtre"
    window   :=   NSWindow   alloc   initWithContentRect:(125<>513
extent:400<>200)
              styleMask:NSTitledWindowMask + NSClosableWindowMask

NSResizableWindowMask
              backing:NSBackingStoreBuffered
              defer:NO.

     "Envoie de la fenêtre sur l écran"
    window orderFront:nil.

     "Donnez un titre à la fenêtre"
    window setTitle:'Convertisseur de devises'.

     "Instantiation d un objet formulaire"
    form := NSForm alloc initWithFrame:(60<>90 extent:320<>85).

     "Mettez le formulaire dans la fenêtre"
    window contentView addSubview:form.

     "Configuration du formulaire"
    form addEntry:'Taux de change pour $1'.
    form addEntry:'Dollars à Convertir'.
    form addEntry:'Valeur dans l autre devise'.
    form setAutosizesCells:YES.

     "Instantiation d une ligne décorative et affichage dans la fenêtre"
    line := NSBox alloc initWithFrame:(15<>70 extent:370<>2).
    window contentView addSubview:line.
```

```
      "Instantiation d un bouton, affichage dans la fenêtre et
configuration"
    button := NSButton alloc initWithFrame:(250<>20 extent:90<>30).
    window contentView addSubview:button.
    button setBezelStyle:NSRoundedBezelStyle.
    button setTitle:'Convertir'.

     "Instantiation du script qui va effectuer la conversion des devises"
    conversionScript := [(form cellAtIndex:2) setStringValue:(form
cellAtIndex:0) floatValue
    * (form cellAtIndex:1) floatValue. form selectTextAtIndex:0].

     "Mettez le script comme cible du formulaire"
    "Le script va être évalué lorsque l'utilisateur appuie sur Return"
    form setTarget:conversionScript.
    form setAction:#value.

     "Mettez le script comme la cible du bouton"

     "Le script va être évalué lorsque l'utilisateur appuie sur Return"
    button setTarget:conversionScript.
    button setAction:#value.
```

Nous remarquons rapidement plusieurs choses :

- Nous référençons directement les classes Cocoa dans le code: `NSWindow`, `NSForm`, `NSButtonNSBox` sont des classes Cocoa.
- Nous référençons directement les constantes Cocoa : `NSTitledWindowMask`, `NSClosableWindowMask`, ou `NSRoundedBezelStyle`sont des noms symboliques définis par Cocoa.
- Nous référençons directement les API Cocoa. les méthodes telles que `alloc`, `initWithFrame:`, `setTitle:`, etc. font partie des frameworks Cocoa.
- La syntaxe pour envoyer des messages aux objecs est semblable à la syntaxe Objective-C, sauf que nous n'utilisons pas `[` et `]` pour maquer le début et la fin de l'envoi d'un message. Cette similitude n'est pas surprenante puisque la syntaxe pour l'envoi des message dans Objective-C est empruntée à Smalltalk.
- Nous pouvons passer des objets F-Script comme arguments à des méthodes Cocoa. Inversement, nous pouvons récupérer des objetsCocoa objects à partir de ces appels et les utiliser plus tard dans F-Script. Comme vous le savez peut-être, l'utilisation de ce genre d'intégration requière souvent l'utilisation de techniques de correspondance assez délicates. Avec F-Script, les choses sont beaucoup plus simple dans ce sens, puisqu'il n'y a rien à faire correspondre : le modèle objets d'F-Script est le modèle objets de Cocoa.

La vrai conversion de devises est effectuée par un objet créé par l'instruction suivante :

```
    conversionScript := [(form cellAtIndex:2) setStringValue:(form
cellAtIndex:0) floatValue
      * (form cellAtIndex:1) floatValue. form selectTextAtIndex:0].
```

La notation `[...]` crée un objet de la classe`Block` qui représente un bloc de code qui peut-être exécuté plus tard (`Block` est une classe Objective-C fourni par le framework d'F-Script). Dans notre bloc, nous récupérons simplement les valeurs des champs dans l'interface utilisateur, effectuons le calcul (qui ne n'implique qu'une multiplication) et nous remettons le résultat dans un élément de l'IU. Unen fois que c'est fait, nous sélectionnons le texte du premier champ pour imiter le comportement du convertisseur de devises d'Apple.

Plus loin dans ce code, notre objet devient la cible des objets `form` et `button`. Ainsi, il est évalué lorsque l'utilisateur appuie sur Return ou clique sur le bouton.

Le script de conversion de devises, version 2

Ci-dessous se trouve la seconde version du convertisseur de devises. Ce script est plus court car il utilise des fonctionnalités très pratiques d'F-Script que nous avions évité d'utiliser dans la première version par soucis de simplicité. Nous avons aussi supprimé les commentaires (le code parle de lui-même) et réorganisé un peu le programme. L'interface est dorénavant entièrement construite et configurée avant d'être envoyée sur l'écran. Enfin, nous avons rajouter le support pour le système de gestion de la mémoire de Cocoa pour être sûr que les divers objets Cocoa que nous créons sont bien détruits quand nous ne les utilisons plus.

CONVERTISSEUR DE DEVISES F-SCRIPT (version 2)

```
    window := NSWindow alloc initWithContentRect:(125<>513
extent:400<>200)
                    styleMask:NSTitledWindowMask + NSClosableWindowMask

NSResizableWindowMask
                    backing:NSBackingStoreBuffered
                    defer:NO.
    conversionScript := [(form cellAtIndex:2) setStringValue:(form
cellAtIndex:0) floatValue
    * (form cellAtIndex:1) floatValue. form selectTextAtIndex:0].

    form := (NSForm alloc initWithFrame:(60<>90 extent:320<>85))
autorelease.
    form addEntry:@{'Taux de change pour 1$','Dollars à convertir','Valeur
dans l autre monnaie'}.
    form setAutosizesCells:YES;
    setTarget:conversionScript; setAction:#value.
    button := (NSButton alloc initWithFrame:(250<>20 extent:90<>30))
autorelease.
    button setBezelStyle:NSRoundedBezelStyle;
    setTitle:'Convertir'; setTarget:conversionScript; setAction:#value.
    line := (NSBox alloc initWithFrame:(15<>70 extent:370<>2))
autorelease.

    window contentView addSubview:@{form, button, line}.
    window setTitle:'Currency Converter'; orderFront:nil.
```

Une des nouveautés que nous utilisons dans cette version est `";"`, notation qui permet d'imbriquer les messages. Cette notation nous permet d'envoyer plusieurs messages vers un recepteur unique sans avoir à spécifier un recepteur à chaque fois.

Autre chose intéressante, l'instruction :

```
    form addEntry:@{'Taux de change pour 1$','Dollars à convertir','Valeur
dans l autre monnaie'}.
```

Une des fonctionnaliltés innovantes d'F-Script est sa capacité à manipuler des groupes entiers d'objets d'un seul coup, et ce, même avec des méthodes qui n'ont pas été spécifiquement définis pour supporter les collecteurs d'objets (en fait, F-Script fourni un langage complet de requêtes objets utilisables directement sur les objets Cocoa).

C'est notamment le cas ici, où nous ajoutons d'un coup une liste complète d'entrée au formulaire. Nous utilisons la notation "message pattern" (denoté par `@`) qui nous permet de spécifier des envois de messages potentiellement complexes. Un "message pattern" implique généralement un ou plusieurs collecteurs d'objets : Dans notre exemple, nous utilisons un tableau d'objets 'chaine de caractères' dénoté par `{` et `}`. Lors de l'exécution, l'instruction va déclencher la génération de ces trois envois de message :

```
form addEntry:'Taux de change pour 1$'
form addEntry:'Dollars à convertir'
form addEntry:'Valeur dans l autre monnaie'
```

Le même shéma est employé dans l'instruction `"window contentView addSubview:@{form, button, line}"` où nous mettons d'un seul coup, un jeu complet de vues dans une fenêtre.

Dans ces exemples, nous utilisons un shéma relativement simple. F-Script fournit un syntaxe complete qui rend possible l'expression d'une vaste variété de "message patterns". Tous les concepts spécifics à F-Script, tels que les "message patterns", peuvent être utilisés quand vous scriptez Cocoa.

Le script de conversion de devises, version 3

Dans l'état actuel, notre script n'est pas très modulaire : C'est juste un jeu d'instructions qui utilise des variables globales et que nous collons dans la console d'F-Script afin de les exécuter. La version suivante introduit la modularité :

- Notre script lui-même va devenir un objet.
- Il va dorénavent utiliser des variables locales au lieu des variables globales de telle sorte que nous ne polluerons plus l'environnement du niveau supérieur.
- Il va prendre le titre de la fenêtre comme argument.

CONVERTISSEUR DE DEVISES F-SCRIPT (version 3)

```
converter := [:title | |window conversionScript form button line|
        window := NSWindow alloc initWithContentRect:(125<>513
extent:400<>200)

NSClosableWindowMask

NSResizableWindowMask
                        backing:NSBackingStoreBuffered
                        defer:NO.
        conversionScript := [(form cellAtIndex:2) setStringValue:(form
cellAtIndex:0)
          floatValue * (form cellAtIndex:1) floatValue. form
selectTextAtIndex:0].

        form := (NSForm alloc initWithFrame:(60<>90 extent:320<>85))
autorelease.
          form addEntry:@{'Taux de change pour 1$','Dollars à
convertir','Valeur dans l autre monnaie'}.
           form setAutosizesCells:YES; setTarget:conversionScr
setAction:#value.
       button := (NSButton alloc initWithFrame:(250<>20 extent:90<>30))
autorelease.
      button setBezelStyle:NSRoundedBezelStyle; setTitle:'Convertir';
      setTarget:conversionScript; setAction:#value.
        line := (NSBox alloc initWithFrame:(15<>70 extent:370<>2))
autorelease.
```

```
    window contentView addSubview:@{form, button, line}.
    window setTitle:title; orderFront:nil.
]
```

Dans cette version, nous avons mis les instructions entre un `[` et un `]` pour faire générer un objet `Block` à F-Script. Nous avons également déclaré les arguments et les variables locales du script sur la première ligne. Enfin, nous donner le nom de "converter" à l'objet `Block` qui représente notre script.

Nous pouvons maintenant invoquer notre script en envoyant un `value` message sans oublier de fournir l'argument requis. Par exemple :

```
converter value:'Ce convertisseur est sympa !'
```

Chaque fois que nous l'invoquons, un nouveau convertisseur complètement fonctionnel est créé et affiché à l'écran.

Comme notre script est un objet de la classe `Block`, nous pouvons le manipuler comme tout autre objet : le mettre dans un collecteur, le passer comme argument à d'autres méthodes, l'archiver sur le disque, et bien plus. Il y a également plusieurs utilitaires spécifiquement liés aux objets `Block`, parmi lesquels un éditeur graphique de code.

Enfin, puisque F-Script est distribué sous la forme d'un framework prêt à être inclus dans toute application Cocoa, il devient très facile de placer notre script dans un executable Mac OS X standard.

Réflexions finales

Nous avons vu les implications qu'apporte le scripting de Cocoa avec F-Script. Il est clair que de nombreux sujets, tels que le traitement des exceptions Cocoa, l'archivage des objets, l'intégration avec Interface Builder, l'utilisation de classe Objective-C personnalisées ou le mappage des types non-objets, n'ont pas été traités dans cet article. Mais vous pouvez vous attendre à une intégration de très haut niveau de ses aspects.

Textes originaux en anglais sur O'Reilly Network : http://www.oreillynet.com/pub/au/767 par Philippe Mouguin.